第33卷 第12期 计 算 机 工 程 2007年6月
Vol.33 No.12 Computer Engineering June 2007
·博士论文· 文章编号：1000—3428(2007)12—0028—03 文献标识码：A 中图分类号：TP316.2

容错调度算法中反向调度与正向调度性能分析

刘 东，张春元
(国防科技大学计算机学院，长沙 410073)

摘 要：分析了软件容错模型中的BCE容错调度算法，针对该算法中的反向调度和正向调度两个过程，给出了RMB、DMB、EDFB 3种反向调度算法和RMF、EDFF 2种正向调度算法，指出了反向调度和正向调度相互协调的特性。将各种算法在BCE算法中进行模拟，结果表明EDFF正向调度算法能够与3种反向调度算法更好地协调，从而获得比RMF正向调度算法更高的调度性能。模拟结果表明，3种反向调度算法在BCE算法中的性能相近。得出RMB(或DMB)反向调度算法与EDFF正向调度算法的组合较适用于软件容错模型的结论。
关键词：容错调度；实时系统；软件容错

Analysis of Backward Schedule and Forward Schedule in Fault-tolerant Schedule Algorithm

LIU Dong, ZHANG Chunyuan
(Department of Computer, National University of Defense Technology, Changsha 410073)

【Abstract】 BCE algorithm and its processes of backward schedule and forward schedule are analyzed. Two backward schedule algorithms, RMF and EDFF, and three forward schedule algorithms, RMB, DMB and EDFB, are researched. The cooperation between backward schedule and forward schedule is put forward. Different schedule algorithms are simulated with BCE algorithm. Since EDFF forward schedule algorithm cooperates well with three backward schedule algorithms, it gets better schedule performance than RMF. Simulation results also show that three backward schedule algorithms have similar effect on BCE algorithm. The conclusion that the combination of EDFF and RMB, or DMB, is more applicable for software fault-tolerant module is made.
【Key words】 Fault-tolerant schedule; Real-time system; Software fault-tolerance

1 概述

为了提高多任务实时系统的可靠性，需要保证系统在任务发生错误时仍旧能够正常运行。针对此问题，目前已经提出了多种容错模型，通过出错任务的再次运行或替代任务的运行来完成实时系统的容错。软件容错模型[1~3]为每个任务提供主部分和替代部分两个版本，正常情况下系统运行功能较完善的主部分，并为替代部分预留出足够的时间资源，当主部分执行失败时，执行只具有基本功能的替代部分。

Ching-Chih等人为截止期机制的软件容错模型提出了BCE调度算法[1]。BCE利用Basic算法作为基本调度算法，首先将替代部分以静态优先级算法进行反向调度，然后在剩余的时间间隔中，利用静态调度或动态调度算法调度主部分的执行。BCE利用CAT算法检查待执行主部分是否有足够的执行时间，从而判断是否取消主部分的执行。利用EIT算法消除空闲的时间间隔，为后续任务留出更多的执行空间。BCE算法在处理器利用率较低时能够获得较好的调度性能，但当处理器利用率较高时，由于不能够利用CAT算法获得较好的主部分可执行预测，因此具有较高的主部分丢失率。

韩建军等人在文献[2,3]中改进了BCE算法，提出PKSA(EBPA)和CUBA算法。两种算法通过对主部分可执行性进行试探性检测，进一步提高了预测的准确性，从而提高了主部分的完成率，并有效地减少了浪费的CPU时间，但两种算法增加了容错调度中判断主部分可执行性的复杂度。

截止期机制的软件容错模型包含反向调度和正向调度两个步骤。反向调度算法在一个计划周期内以从后至前的方式调度替代部分的执行时间，使替代部分尽可能地靠后执行。正向调度算法在反向调度的基础上以从前至后的方式调度主部分的执行。反向调度的目的是为了给正向调度提供更好的调度空间，因此两种调度算法是相互协调的过程。两个调度过程均可以采用多种调度算法，而在已有的研究中，均是直接采用某种反向调度算法和正向调度算法，而缺少对各种调度算法之间的比较分析，对各种调度算法在软件容错模型中的适应性较少涉及。由于BCE算法具有复杂度适中，调度性能较好的优点，本文将在BCE算法的基础上研究各种反向调度算法和正向调度算法的调度性能，从而判定各种算法对容错调度的适用性。

2 截止期机制的软件容错模型

在截止期机制的软件容错模型中，实时系统包含周期性的任务集合$\tau=\{\tau_1,\tau_2,\ldots,\tau_n\}$，每个任务$\tau_i$的周期为$T_i$。$\tau_i$的每次执行称为一个请求。$\tau_i$的第j个请求用$J_{ij}$表示，其中$1\le i\le n$，$j\ge 1$。定义$r_{ij}=(j-1)$，$T_i$为$J_{ij}$的释放时间，$d_{ij}=jT_i$为$J_{ij}$的截止期，则$J_{ij}$必须在$d_{ij}$之前完成。定义计划周期T为所有n个任务周期的最小公倍数，即$T=LCM(T_1,T_2,\ldots,T_n)$。由于任务每经过一个计划周期将会重复执行，因此不失一般性，仅考虑所有任务

作者简介：刘 东(1981－)，男，博士生，主研方向：计算机体系结构，实时系统；张春元，教授、博导
收稿日期：2006-08-05 **E-mail：**windleaf1980.163.com

在一个计划周期内的调度情况。设所有任务从 0 时刻开始执行。

每个任务τ_i具有两个独立的程序版本：主部分P_i和替代部分A_i。P_i的执行时间为p_i，A_i的执行时间为a_i，通常$p_i \geq a_i$，则$\tau_i = (T_i, p_i, a_i)$可以唯一确定任务τ_i。相应的，τ_i的第j个请求J_{ij}包含主部分P_{ij}和替代部分A_{ij}。定义FP为软件错误概率，表示任务主部分执行完毕后发生错误的概率。

在计划周期内，τ_i的每个请求J_{ij}必须在其截止期d_{ij}内完成主部分P_{ij}或替代部分A_{ij}。A_{ij}的开始时间称为通知时间v_{ij}，当P_{ij}在v_{ij}之前全部完成，则撤销A_{ij}的执行，否则，撤销P_{ij}的执行，转而开始执行A_{ij}。主部分包含了更多的功能，如果执行成功，将获得更佳的操作结果；但由于其复杂度高，难以测试和验证，因此可靠性难以保证。替代部分只具有最简单的功能，只产生最基本的可接受操作结果，但由于其易于测试和可靠性验证，因此被认为能够完全正确地执行。由于主部分具有更高的结果质量，因此调度算法应当尽可能地调度更多的主部分完成。

3 调度算法

3.1 反向调度算法

反向调度是在正向调度主部分之前，将替代部分以从后至前的方式安排好替代部分的执行时间，使替代部分在时间轴上尽可能地靠后执行，为主部分留出更多的调度空间。反向调度可以采用多种调度算法，文献[1]利用任务周期作为反向调度的静态优先级，周期短的任务具有较高的优先级，文献[2,3]提出容错截止期的概念，容错截止期小的任务具有较高的静态优先级。在此类静态调度过程中均未考虑替代部分截止期的因素，本文在此提出替代部分反向截止期的概念，并提出两种基于反向截止期的反向调度算法。

定义在从T到 0 的反向调度过程中，替代部分A_{ij}的截止期为反向截止期，用BD_{ij}表示。

在反向调度时，调度后的A_{ij}必须能够保证同一周期内余下的时间间隔足够P_{ij}使用，即A_{ij}必须在某一截止期之前调度完成，本文称之为A_{ij}的反向截止期BD_{ij}。如图 1 所示，假设P_{ij}在$(j-1)T_i$开始执行，如果A_{ij}的通知时间v_{ij}小于P_{ij}的最早完成时间$(j-1)T_i+p_i$，则P_{ij}将无法获得足够的执行时间，从而不能顺利完成。

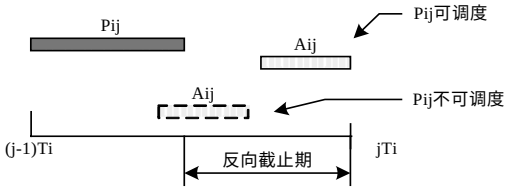


图 1 反向截止期

未考虑反向截止期的调度均采用RM算法[4]，本文称之为RMB(RM backward)算法，如果反向调度过程中考虑反向截止期这一因素，则可采用静态和动态调度算法。由于DM和EDF算法已分别被证明是最优的考虑截止期的静态和动态调度算法，本文将采用这两种方式的反向调度算法，分别称为DMB(DM backward)算法和EDFB(EDF backward)算法。

在DMB算法中，以T_i与p_i之差作为静态的反向截止期，即$BD_{ij} = T_i - P_i$，具有较小反向截止期的A_{ij}分配较高的静态优先级。在EDFB算法中，将主部分的到达时间$(j-1)T_i$与执行时间p_i之和作为动态的反向截止期，即$BD_{ij} = (j-1)\ T_i+p_i$，具有较大反向截止期的A_{ij}分配较高的动态优先级。

DMB和EDFB算法的可行性由FC(feasibility checking)算法[5]判定。

3.2 正向调度算法

正向调度是在反向调度的基础上调度主部分的执行，并可以采用多种调度算法。文献[1]提出正向调度可以采用静态或者动态调度算法，并在实际模拟过程中采用了静态的 RM 算法，称之为 RMF(RM forward)算法。文献[2,3]在模拟过程中采用基于通知时间的动态调度算法，通知时间即为主部分的截止期，因此称该算法为 EDFF(EDF forward)算法。

3.3 正向调度与反向调度的协调

由于反向调度和正向调度具有多种调度方法，因此其各种调度方法的组合会给最终的实时容错调度算法带来性能的差别。

考虑任务集$\tau=\{\tau_1,\tau_2\}=\{(T_1,p_1,a_1),(T_2,p_2,a_2)\}=\{(10,4,3),(8,3,2)\}$，假设零时刻$P_{11}$和$P_{21}$同时开始准备执行。

若反向调度和正向调度分别采用RMB和EDFF算法，并认为所有主部分均不发生错误，则算法对各个任务的调度结果如图 2(a)所示。在这种情况下，由于P_{21}没有足够的执行时间，因此执行失败。

若反向调度和正向调度分别采用EDFB和EDFF算法，则调度结果如图 2(b)所示，其中，主部分P_{11}和P_{21}均可调度完毕。

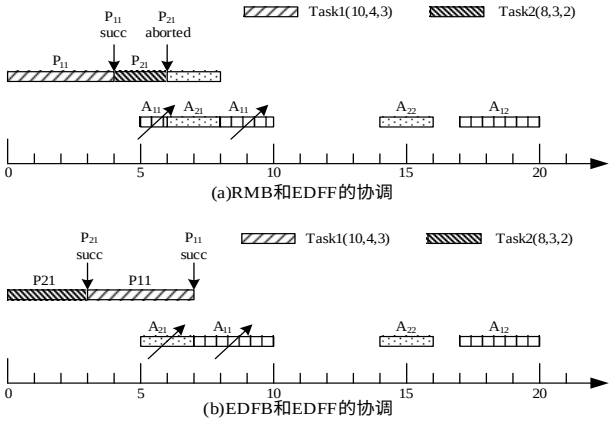


图 2 静态优先级反向调度对任务调度结果的影响

在上述示例中，EDFB反向调度算法引入反向截止期的概念，兼顾主部分的执行特征，从而为随后的EDFF正向调度算法中的主部分留出了更多的调度空间，获得了较好的调度性能。但如果对上述任务集采用RMB和RMF算法，则同样能够将P_{11}和P_{21}调度完毕。因此反向调度算法和正向调度算法的协调对最终的任务调度结果会产生影响，两种调度算法具有相互协调的特性。

4 模拟结果

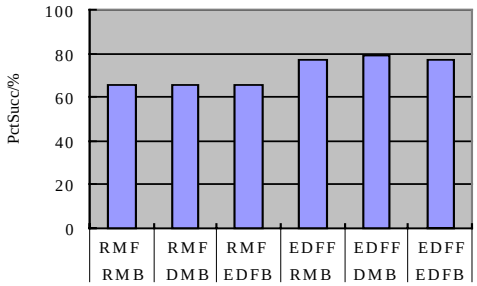
4.1 评价标准

本文采用主部分完成率 PctSucc 和浪费的 CPU 时间 W 作为最终调度结果的评价标准。主部分完成率 PctSucc 是指所有任务在计划周期内执行成功的主部分数目与未发生错误的主部分数目的比例，PctSucc 衡量了调度算法的调度性能。若主部分因执行时间不够而被取消，此时主部分已经占用的 CPU 时间称为浪费的 CPU 时间 W，该参数表示了 CAT 算法预测的准确性。

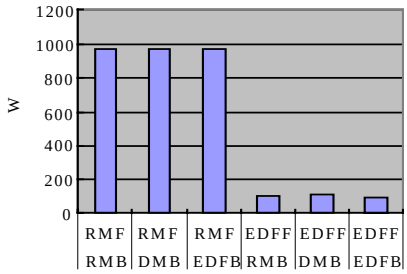
4.2 RMF 与 EDFF 的比较

任务的错误概率FP=0.1 时，考虑任务集$\tau=\{\tau_1, \tau_2, \tau_3, \tau_4\}=\{(15,3,1),(16,4,3),(30,8,4),(31,10,6)\}$，则任务集的计划周期

为T = LCM(15,16,30,31) = 7 440。该任务集的特点是，在进行反向调度时，各种反向调度算法所获得的反向调度优先级不同。各种调度算法在上述任务集的模拟结果如图 3 所示。



(a)两种正向调度算法下获得的 PctSucc



(b)两种正向调度算法下获得的 W

图 3 RMF 与 EDFF 在 BCE 算法中的比较

模拟结果表明：当正向调度采用RMF算法时，无论采用何种反向调度算法，所获得的主部分完成率均为 66%；这是由于RMF算法只关注任务周期，而忽略事先已分配的替代部分所占用的时间片断，因此反向调度算法对RMF算法几乎没有影响。当正向调度采用EDFF算法时，3 种调度算法获得的性能相近，DMB算法相对于RMB和EDFB算法能够获得稍好的调度性能。此外，RMF算法较EDFF算法性能低，这是由于每一个主部分均具有相应的截止期(替代部分的通知时间v_{ij})，而v_{ij}并不是一个常数，因此考虑截止期的基于动态优先级的EDFF算法更适合实际的任务调度，从而体现了正向调度与反向调度相互协调的特性。

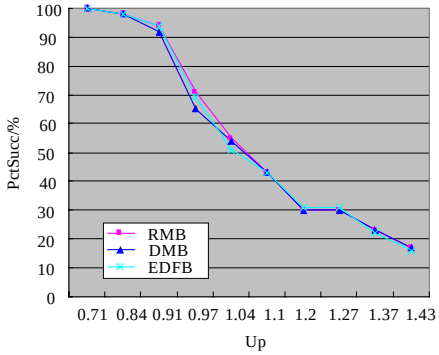
4.3 RMB、DMB 与 EDFB 的比较

由于 EDFF 算法较 RMF 算法能够获得较佳的调度性能，因此在比较 RMB、DMB、EDFB 这 3 种反向调度算法时，本文采用 EDFF 正向调度算法。

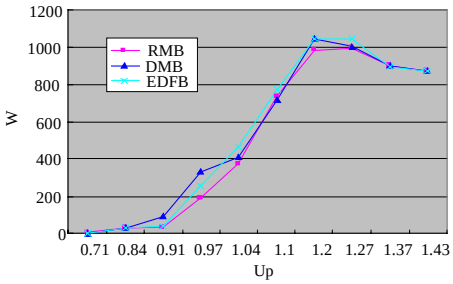
本文在模拟过程中固定任务周期T_i和替代部分执行时间a_i，通过更改各主部分的执行时间p_i达到间接更改实际CPU利用率的目的。因此，本文模拟在不同的主部分CPU利用率U_p的条件下，3 种反向调度算法对最终调度性能的影响。

当FP=0.1 时，考虑任务集$\tau=\{\tau_1,\tau_2,\tau_3,\tau_4\}=\{(15,p_1,1),\ (16,p_2,3),\ (30,p_3,4),(31,p_4,6)\}$，在不同的$U_p$下，RMB、DMB、EDFB 这 3 种反向调度算法的比较结果如图 4 所示。

模拟结果表明，随着U_p的提高，3 种调度算法的性能也逐渐降低。这是由于主部分执行时间的增加导致可供调度的空闲时间减小，将会有更多的主部分在完成工作之前即到达通知时间，从而造成执行失败。在模拟过程中，不同的反向算法所获得的调度性能相近，这是由于BCE算法中的CAT算法能够预测主部分能否成功，尤其当任务集的数目较少时，这种预测具有较高的准确率。因此，替代部分的时间分配对具有动态预测功能的 BCE 算法产生较小影响，而且，EDFF 算法均能与 3 种反向调度算法较好地协调工作。在本次模拟中，RMB 算法相对于其他两种算法能够取得稍高的主部分完成率和较低的 CPU 浪费，而在前文的模拟结果中，DMB 算法的调度性能优于其他两种算法，因此反向调度算法对最终的 BCE 算法性能影响取决于任务集本身的特性。



(a) 3 种反向调度算法下获得的 PctSucc



(b) 3 种反向调度算法下获得的 W

图 4 RMB、DMB 与 EDFB 在 BCE 算法中的比较

5 结束语

本文探讨了截止期机制的软件容错模型中，RMF 和 EDFF 2 种正向调度算法和 RMB、DMB、EDFB 3 种反向调度算法对容错调度算法的影响。经过模拟分析，EDFF 算法较 RMF 算法具有更高的调度性能，而在 EDFF 正向调度算法中，3 种反向调度算法在 BCE 算法中性能相近。由于 RMB 和 DMB 算法具有较小的复杂度，因此更加适用于软件容错调度模型。

参考文献

1 Ching-Chih H, Shin K G, Wu J. A Fault-tolerant Scheduling Algorithm for Real-time Periodic Tasks with Possible Software Faults[J]. IEEE Transactions on Computers, 2003, 52(3): 362-372.

2 韩建军, 李庆华, Essa A A. 基于软件容错的动态实时调度算法[J]. 计算机研究与发展, 2005, 42(2): 315-321.

3 李庆华, 韩建军, Essa A A, et al. 硬实时系统中基于软件容错的动态调度算法[J]. 软件学报, 2005, 16(1): 101-107.

4 Liu C L, Layland J W. Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment[J]. Journal of the ACM, 1973, 20(1): 46-61.

5 Lin T H, Tarng W. Scheduling Periodic and Aperiodic Tasks in Hard Real-time Computing Systems[J]. ACM Performance Evaluation Review, 1991, 19(1): 31-38.